阿里云
云数据库 POLARDB

性能白皮书

文档版本：20190819

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 POLARDB for MySQL性能白皮书

本文档介绍如何使用Sysbench 0.5测试POLARDB for MySQL集群的最大性能。

背景信息

SysBench是一个跨平台且支持多线程的模块化基准测试工具，用于评估系统在运行高负载的数据库时相关核心参数的性能表现。SysBench的目的是为了绕过复杂的数据库基准设置，甚至在没有安装数据库的前提下，快速了解数据库系统的性能。

准备工作

- 三台ECS实例：
  - 每台有32个vCPU，例如规格ecs.sn1ne.8xlarge。
  - ECS实例与POLARDB for MySQL集群位于同一个地域的同一个可用区。
  - 已设置ECS实例的账号密码。
  - 操作系统为CentOS 7.4 64位。
- 一个POLARDB for MySQL集群：
  - 其中包含一个主实例和一个只读实例。（测试多只读实例时，需要多个只读实例。）
  - 已设置集群的账号和密码。
  - 已在白名单中添加ECS实例的内网IP地址。

> 说明：
> 2核4GB的集群为入门级，用于测试、体验和极小负载的场景，高负载的生产环境不建议使用。生产环境推荐使用8核32GB或以上规格的集群。

安装测试工具

1. **在ECS中执行如下命令安装sysbench 0.5。**

   ```
   yum install gcc gcc-c++ autoconf automake make libtool bzr mysql-devel
    git mysql

   git clone https://github.com/akopytov/sysbench.git

   cd sysbench

   git checkout 0.5

   ./autogen.sh

   ./configure --prefix=/usr --mandir=/usr/share/man

   make

   make install
   ```

2. **执行如下命令配置Sysbench client，使内核可以使用所有的CPU核处理数据包（默认设置为使用2个核），同时减少CPU核之间的上下文切换。**

   ```
   sudo sh -c 'for x in /sys/class/net/eth0/queues/rx-*; do echo ffffffff
   >$x/rps_cpus; done'
   ```

   注：ffffffff表示使用32个核。请根据实际配置修改，例如，如果ECS为8核，则输入ff。

   ```
   sudo sh -c "echo 32768 > /proc/sys/net/core/rps_sock_flow_entries"

   sudo sh -c "echo 4096 > /sys/class/net/eth0/queues/rx-0/rps_flow_cnt"

   sudo sh -c "echo 4096 > /sys/class/net/eth0/queues/rx-1/rps_flow_cnt"
   ```

测试方法

1. 获取POLARDB for MySQL集群的主地址和端口。
   a) 登录POLARDB控制台，进入集群列表页面。
   b) 单击集群ID。
   c) 在基本信息页面的访问信息中找到集群的主地址和端口，如下图所示。

2. 在ECS上执行如下命令，以在POLARDB集群中创建数据库sbtest。

```
mysql -h XXX -P XXX -u XXX -p XXX -e 'create database sbtest'
```

> **说明:**
> 请将本命令和后续步骤的命令中的XXX替换为POLARDB集群的主地址、端口号、用户名和密码。

3. 准备测试数据：用Sysbench在数据库上创建表并插入数据。

```
sysbench --test=sysbench/tests/db/oltp.lua --mysql-host=XXX --mysql
-port=XXX --mysql-user=XXX --mysql-password=XXX --mysql-db=sbtest
 --mysql-table-engine=innodb --oltp-table-size=25000 --oltp-tables-
count=250 --db-driver=mysql prepare
```

4. 使用sysbench测试数据库的读性能，将持续10分钟。

```
sysbench --test=sysbench/tests/db/oltp.lua --mysql-host=XXX --oltp-
tables-count=250 --mysql-user=XXX --mysql-password=XXX --mysql-port=
XXX --db-driver=mysql --oltp-tablesize=25000 --mysql-db=sbtest --max
-requests=0 --oltp_simple_ranges=0 --oltp-distinct-ranges=0 --oltp-
```

```
sum-ranges=0 --oltp-order-ranges=0 --max-time=600 --oltp-read-only=
on --num-threads=500 run
```

5. **使用sysbench测试数据库的写性能，将持续10分钟。**

```
sysbench --test=sysbench/tests/db/oltp.lua --mysql-host=XXX --oltp-
tables-count=250 --mysql-user=XXX --mysql-password=XXX --mysql-port
=XXX --db-driver=mysql --oltp-tablesize=25000 --mysql-db=sbtest --
max-requests=0 --max-time=600 --oltp_simple_ranges=0 --oltp-distinct
-ranges=0 --oltp-sum-ranges=0 --oltp-order-ranges=0 --oltp-point-
selects=0 --num-threads=128 --randtype=uniform run
```

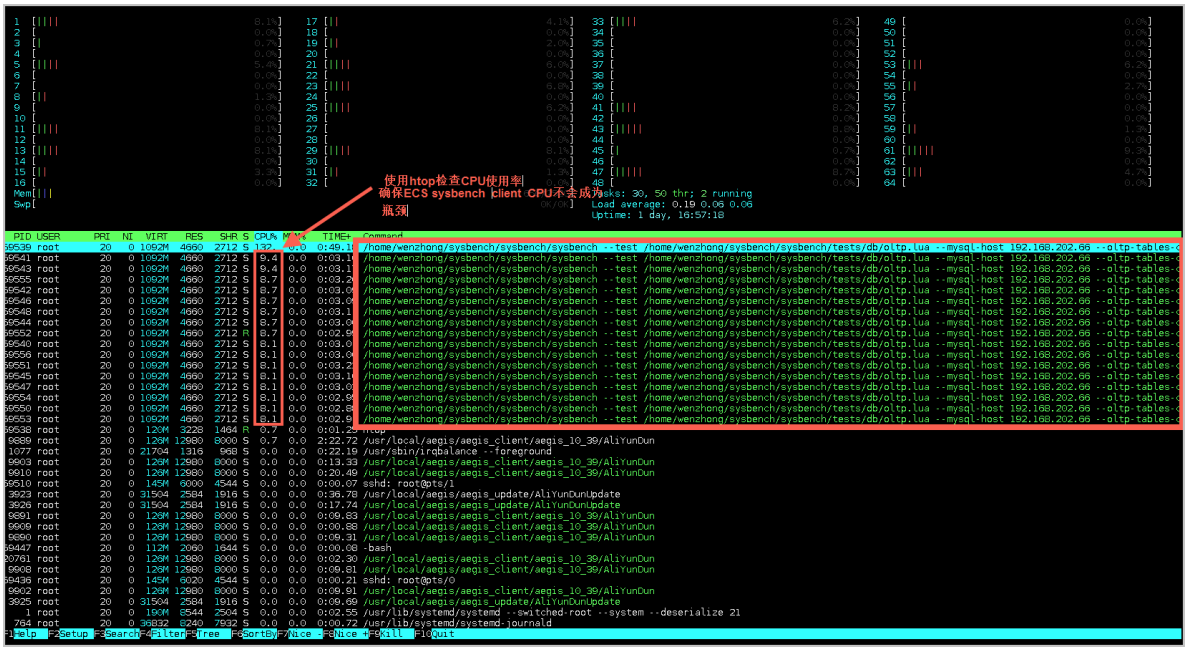
6. **在以上测试的过程中，您可以另外打开一个会话窗口，连接该ECS实例，并使用htop查看sysbench client的CPU使用率是否正常。**

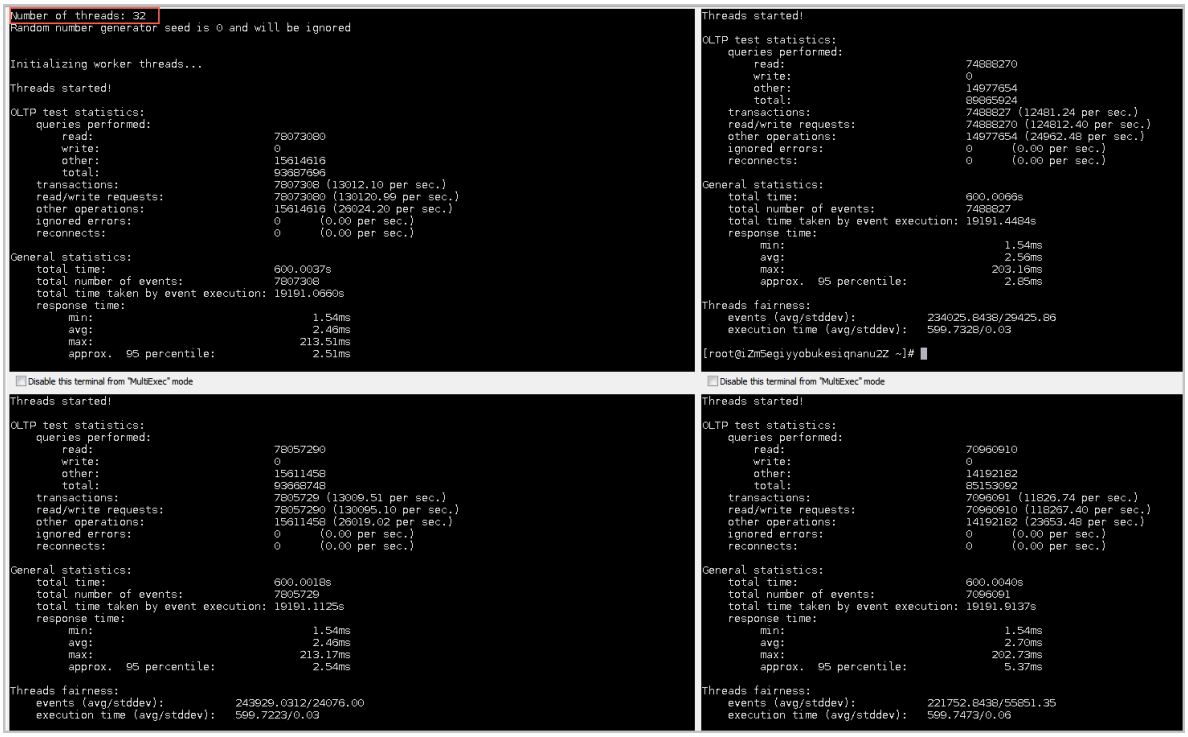
```
yum install htop

htop
```

说明:

- **执行htop后，可以按Q键退出。**

- 关于htop详解，请参见Htop官网。





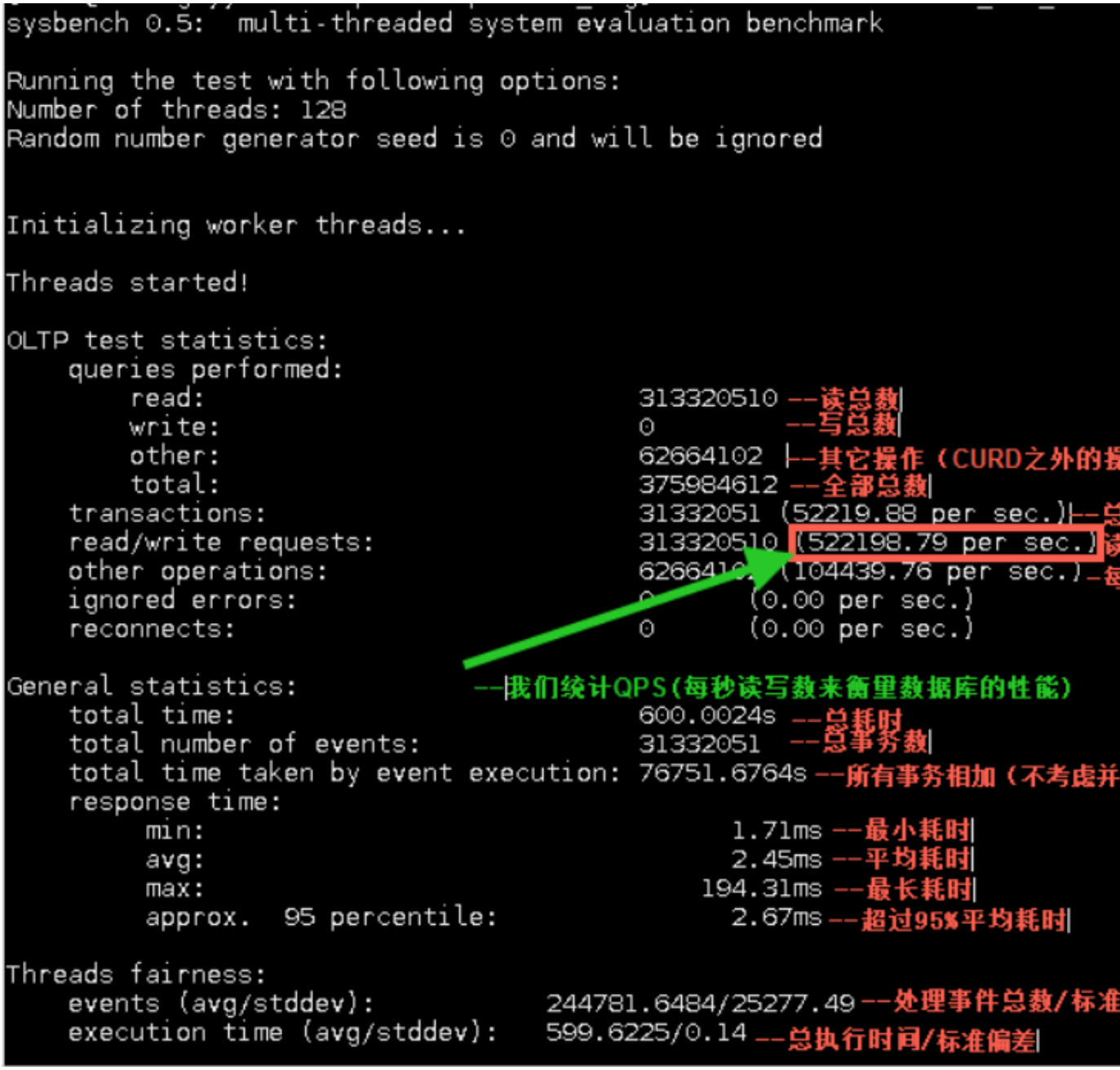
预期结果

使用三台ECS进行数据库性能测试，从log中取测试结果中的QPS和TPS，如下图所示。

说明:

QPS（Queries Per Second）表示数据库每秒执行的SQL数（含INSERT、SELECT、UPDATE、DELETE等）。

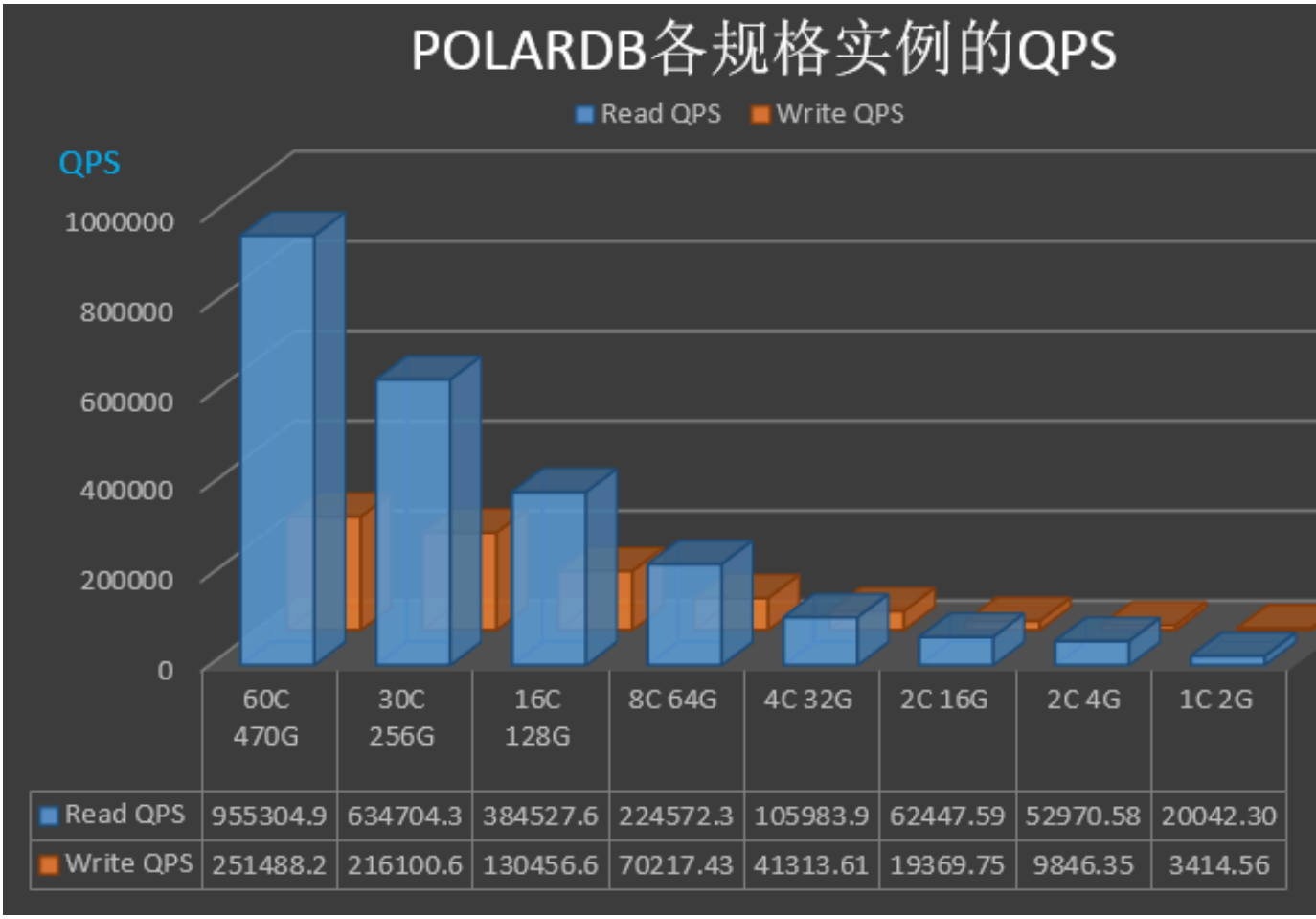
单台ECS的log结果如下：



三台ECS实例通过集群连接串连接一个POLARDB for MySQL集群时，总QPS如下。

说明:

集群连接串总是连接到主实例，所以以下QPS为主实例的性能。

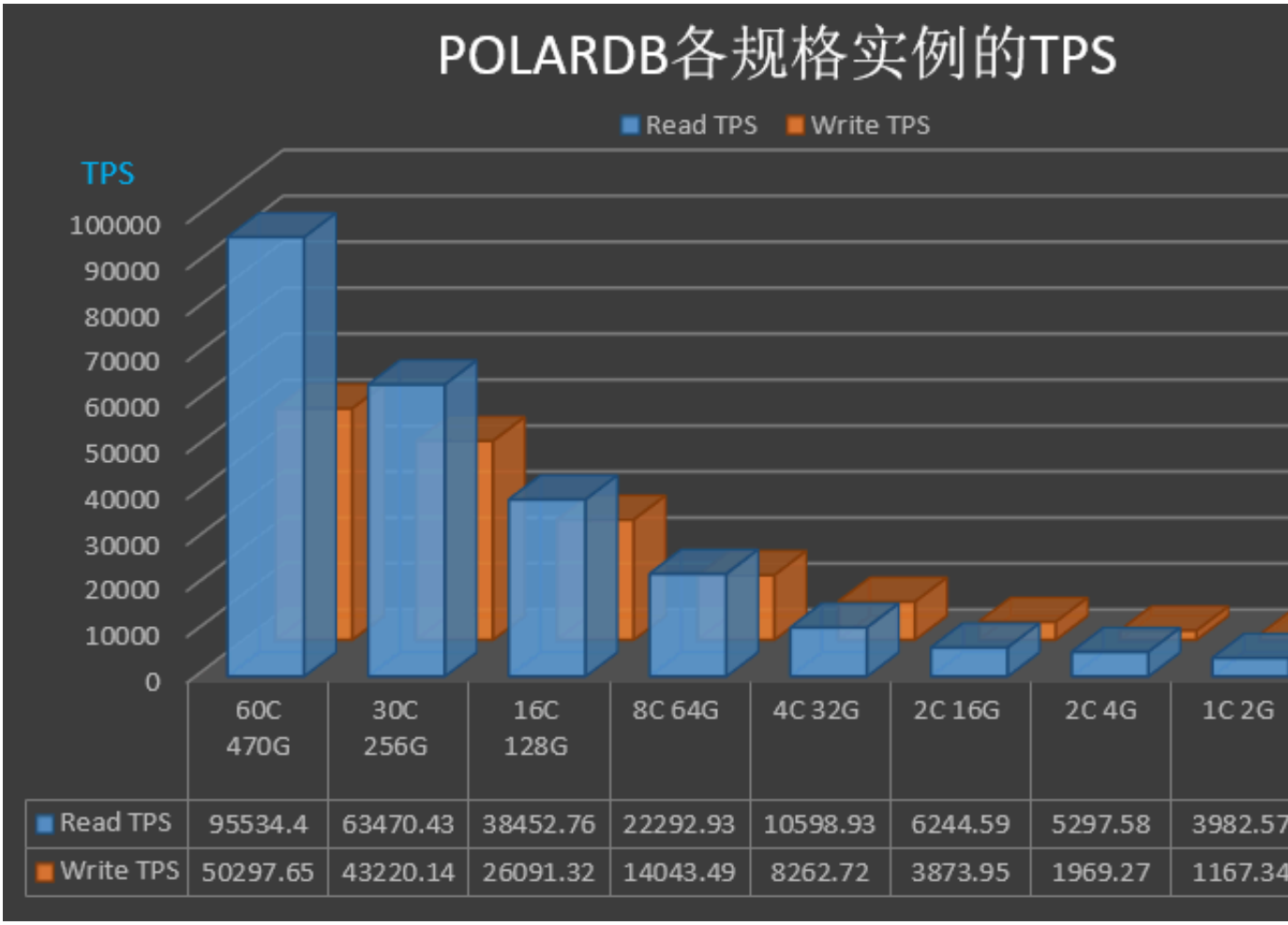


	60C 470G	30C 256G	16C 128G	8C 64G	4C 32G	2C 16G	2C 4G	1C 2G
Read QPS	955304.9	634704.3	384527.6	224572.3	105983.9	62447.59	52970.58	20042.30
Write QPS	251488.2	216100.6	130456.6	70217.43	41313.61	19369.75	9846.35	3414.56

三台ECS实例通过集群连接串连接一个POLARDB for MySQL集群时，总TPS如下。

说明:

集群连接串总是连接到主实例，所以以下TPS为主实例的性能。

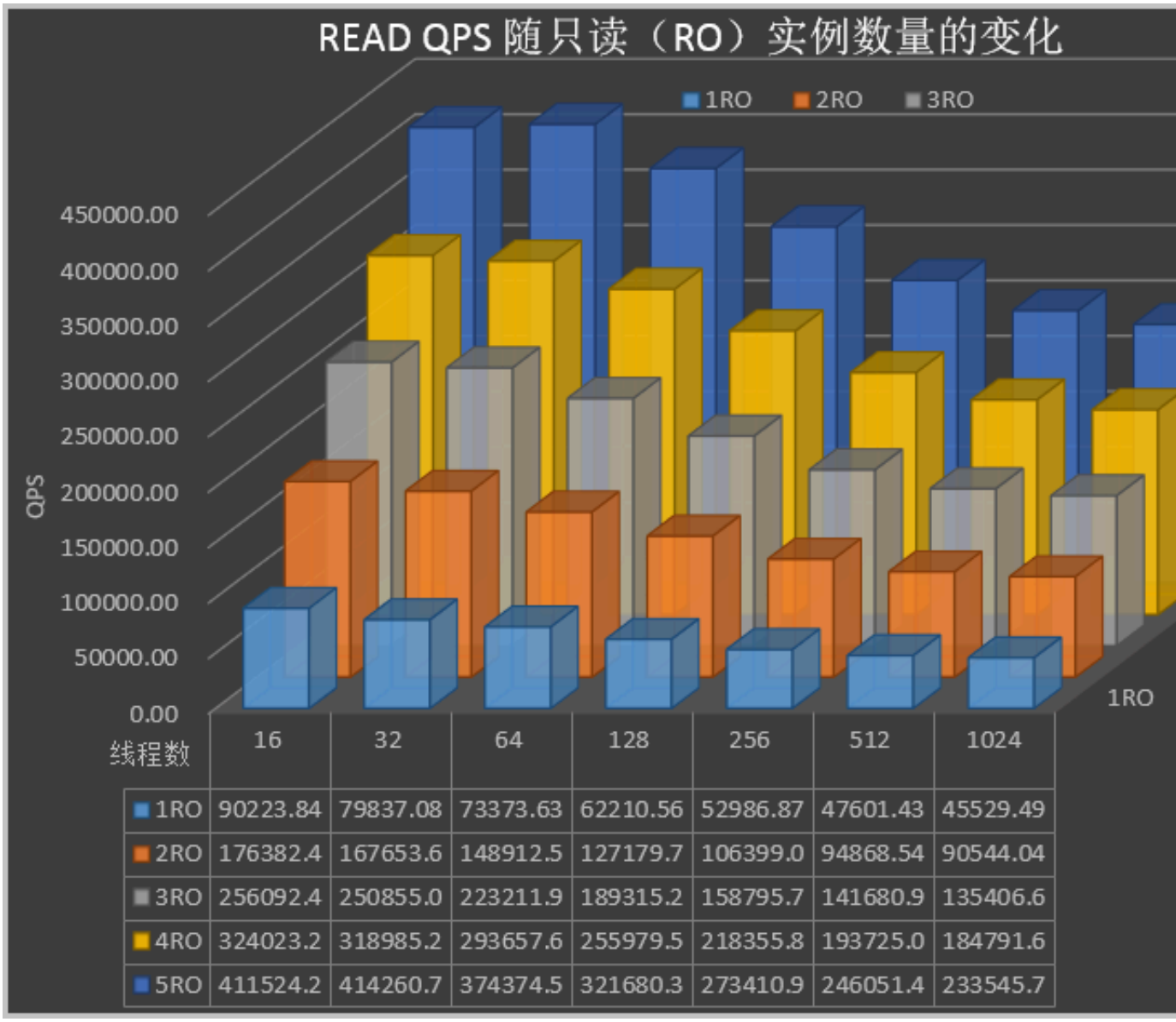


	60C 470G	30C 256G	16C 128G	8C 64G	4C 32G	2C 16G	2C 4G	1C 2G
Read TPS	95534.4	63470.43	38452.76	22292.93	10598.93	6244.59	5297.58	3982.57
Write TPS	50297.65	43220.14	26091.32	14043.49	8262.72	3873.95	1969.27	1167.34

五台ECS实例通过只读实例连接串连接到一个POLARDB for MySQL集群中各个只读实例时，聚合QPS如下。

说明:

- **该集群中包含五个只读实例，每个实例的规格为4C 32G。**
- **每台ECS连接到一个只读实例。**



线程数	16	32	64	128	256	512	1024
1RO	90223.84	79837.08	73373.63	62210.56	52986.87	47601.43	45529.49
2RO	176382.4	167653.6	148912.5	127179.7	106399.0	94868.54	90544.04
3RO	256092.4	250855.0	223211.9	189315.2	158795.7	141680.9	135406.6
4RO	324023.2	318985.2	293657.6	255979.5	218355.8	193725.0	184791.6
5RO	411524.2	414260.7	374374.5	321680.3	273410.9	246051.4	233545.7

2 POLARDB for PostgreSQL性能白皮书

本文档介绍如何使用pgbench测试POLARDB for PostgreSQL集群主节点的最大性能。

PostgreSQL自带一款轻量级的压力测试工具pgbench。pgbench是一种在PostgreSQL上运行基准测试的简单程序，它可以在并发的数据库会话中重复运行相同的SQL命令。

测试环境

- 所有测试均在本地测试完成（使用IP+端口）。
- ECS的实例规格：ecs.g5.16xlarge（64核 256GiB）
- 网络类型：专有网络
- 操作系统：CentOS 7.6 x64

> 说明:
> CentOS 6不支持PostgreSQL 11。

测试指标

- 只读QPS

  数据库只读时每秒执行的SQL数（仅包含SELECT）。

- 读写QPS

  数据库读写时每秒执行的SQL数（包含INSERT、SELECT、UPDATE）。

准备工作

- 安装测试工具

  执行如下命令在ECS实例中安装PostgreSQL 11。

```
yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
yum install -y https://download.postgresql.org/pub/repos/yum/11/
redhat/rhel-7-x86_64/pgdg-centos11-11-2.noarch.rpm
yum install -y postgresql11*
su - postgres
vi .bash_profile
export PS1="$USER@`/bin/hostname -s`-> "
export LANG=en_US.utf8
export PGHOME=/usr/pgsql-11
export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/
lib64:/lib:/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH
export DATE=`date +"%Y%m%d%H%M"`
export PATH=$PGHOME/bin:$PATH:.
export MANPATH=$PGHOME/share/man:$MANPATH
alias rm='rm -i'
alias ll='ls -lh'
```

```
unalias vi
```

- 修改PostgreSQL集群参数

  由于部分参数无法在控制台直接修改，您需要提交工单申请，修改POLARDB for PostgreSQL集群的yaml配置文件如下：

```
default_statistics_target: "100"
max_wal_size: "64GB" #half mem size
effective_cache_size: "96GB" #3/4 mem size
max_parallel_workers_per_gather: "16" #half cpu core number
maintenance_work_mem: "2GB" #1/32 mem, don't exceed 8GB
checkpoint_completion_target: "0.9"
max_parallel_workers: "32" #cpu core number,don't exceed 64
max_prepared_transactions: "2100"
archive_mode: "off"
work_mem: "64MB" #mem 1/2000,don't exceed 128MB
wal_buffers: "16MB"
min_wal_size: "64GB" #1/4 mem size, min size 3GB (3 wal files, 2 as
preallocated)
shared_buffers: "192GB" #75% mem size 8GB
max_connections: "12900"
polar_bulk_extend_size: "4MB"
polar_xlog_record_buffers: "25GB" #10~15% mem size,min size 1GB
hot_standby_feedback: "on"
full_page_writes: "off"
synchronous_commit: "on"
polar_enable_async_pwrite: "off"
polar_parallel_bgwriter_delay: "10ms"
polar_max_non_super_conns: '12800'
polar_parallel_new_bgwriter_threshold_lag: "6GB"
polar_use_statistical_relpages: "on"
polar_vfs.enable_file_size_cache: "on"
```

说明：

以规格为polar.pg.x8.4xlarge（32核 256GB）的集群为例修改配置文件，同时为了和RDS对比，修改内存和RDS for PostgreSQL相同。

修改配置后，重启PostgreSQL集群让配置生效。

测试方法

1. 根据目标库大小初始化测试数据，具体命令如下：
   - 初始化数据50亿：`pgbench -i -s 50000`
   - 初始化数据10亿：`pgbench -i -s 10000`
   - 初始化数据5亿：`pgbench -i -s 5000`
   - 初始化数据1亿：`pgbench -i -s 1000`
2. 通过以下命令配置环境变量：

```
export PGHOST=<PostgreSQL集群主节点私网地址>
export PGPORT=<PostgreSQL集群主节点私网端口>
export PGDATABASE=postgres
```

```
export PGUSER=<PostgreSQL数据库用户名>
export PGPASSWORD=<PostgreSQL对应用户的密码>
```

3. 创建只读和读写的测试脚本。

   · 创建只读脚本ro.sql内容如下：

```
\set aid random_gaussian(1, :range, 10.0)
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
```

   · 创建读写脚本rw.sql内容如下：

```
\set aid random_gaussian(1, :range, 10.0)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid
 = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid
 = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid
 = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES
 (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
END;
```

4. 使用如下命令测试。

   · 只读测试：

```
polar.o.x8.4xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=100000000
polar.o.x8.4xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=500000000
polar.o.x8.4xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=1000000000
polar.o.x8.2xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=100000000
polar.o.x8.2xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=500000000
polar.o.x8.2xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=100000000
polar.o.x4.xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=500000000
polar.o.x4.xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.large，总数据量5亿，热数据1亿
```

```
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=100000000
polar.o.x4.large，总数据量5亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=500000000
polar.o.x4.medium，总数据量1亿，热数据5000万
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=50000000
polar.o.x4.medium，总数据量1亿，热数据1亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=100000000
```

- 读写测试：

```
polar.o.x8.4xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=100000000
polar.o.x8.4xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=500000000
polar.o.x8.4xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=1000000000
polar.o.x8.2xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=100000000
polar.o.x8.2xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=500000000
polar.o.x8.2xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=100000000
polar.o.x4.xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=500000000
polar.o.x4.xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.large，总数据量5亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=100000000
polar.o.x4.large，总数据量5亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=500000000
polar.o.x4.medium，总数据量1亿，热数据5000万
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=50000000
polar.o.x4.medium，总数据量1亿，热数据1亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=100000000
```

说明：

- scale乘以10万：表示测试数据量。
- range：表示活跃数据量。

- -c：表示测试连接数，测试连接数不代表该规格的最大连接数，最大连接数请参考#unique_5。

各规格标准测试结果

规格	测试数据量	热（活跃）数据量	只读QPS	读写QPS
polar.pg.x8.4xlarge 32核 256G	10亿	1亿	522160	294915
polar.pg.x8.4xlarge 32核 256G	10亿	5亿	514143	282645
polar.pg.x8.4xlarge 32核 256G	10亿	10亿	493321	268473
polar.pg.x8.2xlarge 16核 128G	10亿	1亿	256998	156330
polar.pg.x8.2xlarge 16核 128G	10亿	5亿	253937	133125
polar.pg.x8.2xlarge 16核 128G	10亿	10亿	243326	115915
polar.pg.x8.xlarge 8核 64G	10亿	1亿	159323	71820
polar.pg.x8.xlarge 8核 64G	10亿	5亿	155498	58140
polar.pg.x8.xlarge 8核 64G	10亿	10亿	152735	58555

规格	测试数据量	热（活跃）数据量	只读QPS	读写QPS
polar.pg.x4.xlarge 8核 32G	10亿	1亿	129323	64235
polar.pg.x4.xlarge 8核 32G	10亿	5亿	115498	53682
polar.pg.x4.xlarge 8核 32G	10亿	10亿	102735	51555
polar.pg.x4.large 4核 16G	5亿	1亿	75729	48648
polar.pg.x4.large 4核 16G	5亿	5亿	63818	43343
polar.pg.x4.medium 2核 8G	1亿	5000万	34386	21383
pg.x8.medium.2 2核 16G 250G	1亿	1亿	33752	15974

与RDS for PostgreSQL对比测试结果

说明:

- 测试使用的是非标准POLARDB for PostgreSQL规格，修改POLARDB for PostgreSQL主节点内存与RDS for PostgreSQL独享型规格相同，对比其性能。
- 下表的规格为RDS for PostgreSQL的实例规格。

规格	预计默认存储空间可存储数据量（可动态扩容）	测试数据量	热（活跃）数据量	只读QPS	读写QPS
pg.x4.4xlarge.2 32核 128G 2T	100亿	10亿	1亿	462190	254915
pg.x4.4xlarge.2 32核 128G 2T	100亿	10亿	5亿	463176	228440
pg.x4.4xlarge.2 32核 128G 2T	100亿	10亿	10亿	473321	200250
pg.x8.2xlarge.2 16核 128G 2T	100亿	10亿	1亿	256998	156330
pg.x8.2xlarge.2 16核 128G 2T	100亿	10亿	5亿	253937	133125
pg.x8.2xlarge.2 16核 128G 2T	100亿	10亿	10亿	243326	115915
pg.x8.xlarge.2 8核 64G 1T	50亿	10亿	1亿	155014	71820
pg.x8.xlarge.2 8核 64G 1T	50亿	10亿	5亿	159878	58140
pg.x8.xlarge.2 8核 64G 1T	50亿	10亿	10亿	152917	58555

规格	预计默认存储空间可存储数据量（可动态扩容）	测试数据量	热（活跃）数据量	只读QPS	读写QPS
pg.x8.large.2 4核 32G 500G	25亿	5亿	1亿	79429	45110
pg.x8.large.2 4核 32G 500G	25亿	5亿	5亿	76268	36375
pg.x8.medium.2 2核 16G 250G	12.5亿	1亿	5000万	49733	24520
pg.x8.medium.2 2核 16G 250G	12.5亿	1亿	1亿	44093	19880

说明:

- 规格：POLARDB for PostgreSQL的规格代码（提工单调整内存和RDS for PostgreSQL相同）。
- 预计默认存储空间可存储数据量：预计该规格默认存储空间可以存储的记录条数。
- 测试数据量：本轮测试数据的记录条数。
- 热（活跃）数据量：本轮测试的查询、更新SQL的记录条数。
- 只读QPS：只读测试的结果，表示每秒请求数。

- **读写QPS：读写测试的结果，表示每秒请求数。**

图 2-1: 相同内存和CPU，只读时POLARDB和RDS的PostgreSQL引擎对比

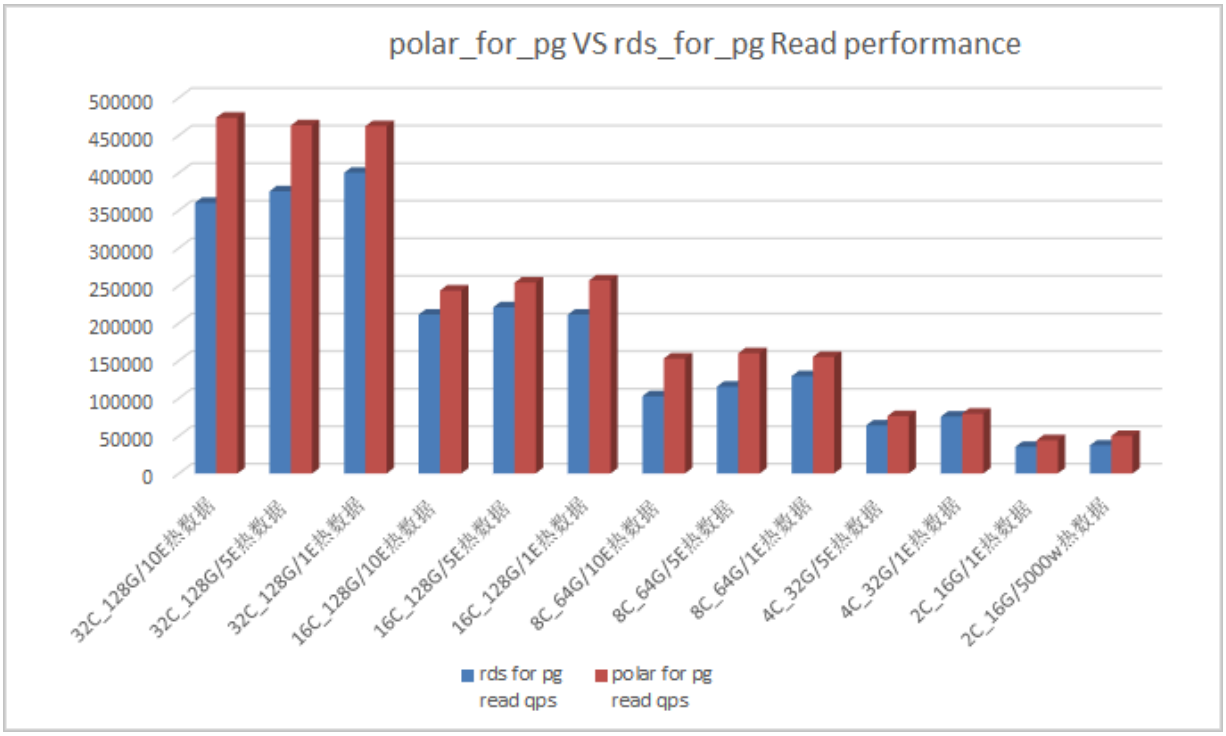
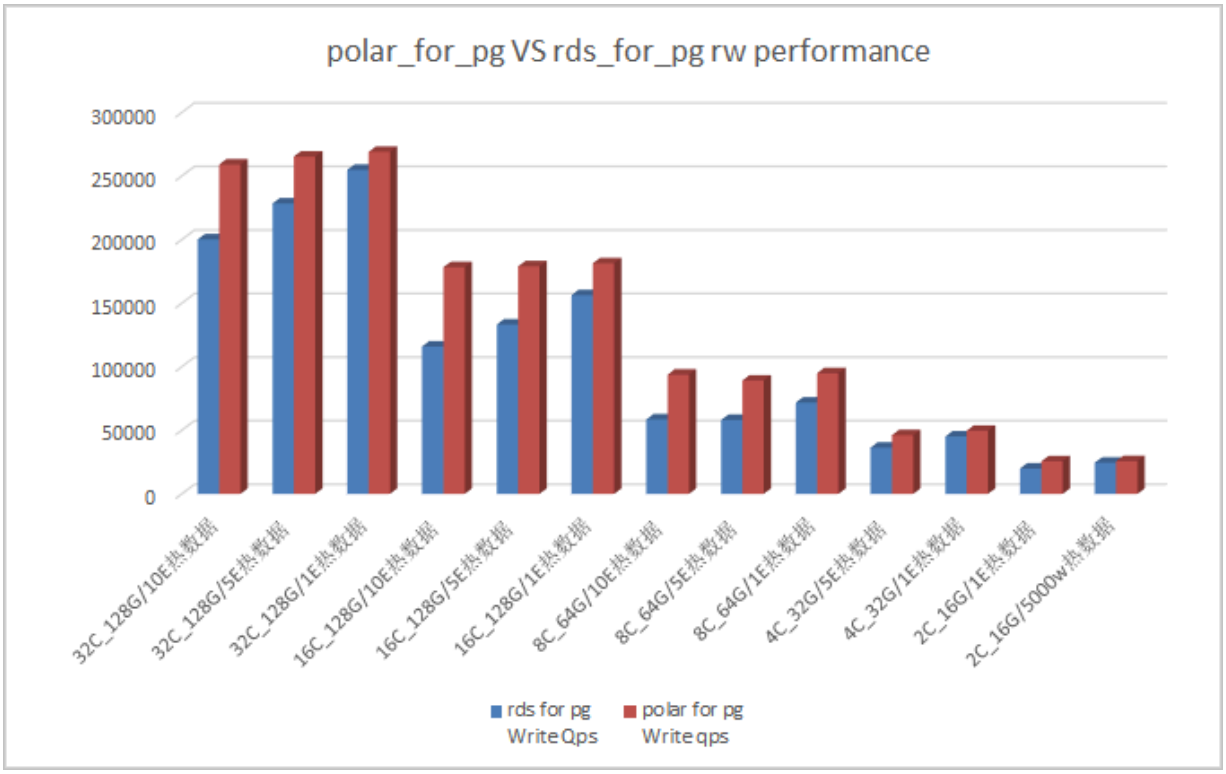


图 2-2: 相同内存和CPU，读写时POLARDB和RDS的PostgreSQL引擎对比



3 POLARDB for Oracle性能白皮书

本文档介绍如何使用pgbench测试POLARDB for Oracle集群主节点的最大性能。

PostgreSQL自带一款轻量级的压力测试工具pgbench。pgbench是一种在PostgreSQL（兼容Oracle）上运行基准测试的简单程序，它可以在并发的数据库会话中重复运行相同的SQL命令。

测试环境

- 所有测试均在本地测试完成（使用IP+端口）。
- ECS的实例规格：ecs.g5.16xlarge（64核 256GiB）
- 网络类型：专有网络
- 操作系统：CentOS 7.6 x64

> 说明:
> CentOS 6不支持PostgreSQL 11。

测试指标

- 只读QPS

  数据库只读时每秒执行的SQL数（仅包含SELECT）。

- 读写QPS

  数据库读写时每秒执行的SQL数（包含INSERT、SELECT、UPDATE）。

准备工作

- 安装测试工具

  执行如下命令在ECS实例中安装PostgreSQL 11。

```
yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
yum install -y https://download.postgresql.org/pub/repos/yum/11/
redhat/rhel-7-x86_64/pgdg-centos11-11-2.noarch.rpm
yum install -y postgresql11*
su - postgres
vi .bash_profile
export PS1="$USER@`/bin/hostname -s`-> "
export LANG=en_US.utf8
export PGHOME=/usr/pgsql-11
export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/
lib64:/lib:/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH
export DATE=`date +"%Y%m%d%H%M"`
export PATH=$PGHOME/bin:$PATH:.
export MANPATH=$PGHOME/share/man:$MANPATH
alias rm='rm -i'
alias ll='ls -lh'
```

```
unalias vi
```

- 修改Oracle集群参数

  由于部分参数无法在控制台直接修改，您需要提交工单申请，修改POLARDB for Oracle集群的yaml配置文件如下：

```
default_statistics_target: "100"
max_wal_size: "64GB" #half mem size
effective_cache_size: "96GB" #3/4 mem size
max_parallel_workers_per_gather: "16" #half cpu core number
maintenance_work_mem: "2GB" #1/32 mem, don't exceed 8GB
checkpoint_completion_target: "0.9"
max_parallel_workers: "32" #cpu core number,don't exceed 64
max_prepared_transactions: "2100"
archive_mode: "off"
work_mem: "64MB" #mem 1/2000,don't exceed 128MB
wal_buffers: "16MB"
min_wal_size: "64GB" #1/4 mem size, min size 3GB (3 wal files, 2 as
preallocated)
shared_buffers: "192GB" #75% mem size 8GB
max_connections: "12900"
polar_bulk_extend_size: "4MB"
polar_xlog_record_buffers: "25GB" #10~15% mem size,min size 1GB
hot_standby_feedback: "on"
full_page_writes: "off"
synchronous_commit: "on"
polar_enable_async_pwrite: "off"
polar_parallel_bgwriter_delay: "10ms"
polar_max_non_super_conns: '12800'
polar_parallel_new_bgwriter_threshold_lag: "6GB"
polar_use_statistical_relpages: "on"
polar_vfs.enable_file_size_cache: "on"
```

说明：

以规格为polar.o.x8.4xlarge（32核 256GB）的集群为例修改配置文件，同时为了和RDS对比，修改内存和RDS for PostgreSQL相同。

修改配置后，重启Oracle集群让配置生效。

测试方法

1. 根据目标库大小初始化测试数据，具体命令如下：
   - 初始化数据50亿：`pgbench -i -s 50000`
   - 初始化数据10亿：`pgbench -i -s 10000`
   - 初始化数据5亿：`pgbench -i -s 5000`
   - 初始化数据1亿：`pgbench -i -s 1000`
2. 通过以下命令配置环境变量：

```
export PGHOST=<Oracle集群主节点私网地址>
export PGPORT=<Oracle集群主节点私网端口>
export PGDATABASE=postgres
```

```
export PGUSER=<Oracle数据库用户名>
export PGPASSWORD=<Oracle对应用户的密码>
```

3. 创建只读和读写的测试脚本。

   · 创建只读脚本ro.sql内容如下：

```
\set aid random_gaussian(1, :range, 10.0)
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
```

   · 创建读写脚本rw.sql内容如下：

```
\set aid random_gaussian(1, :range, 10.0)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid
 = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid
 = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid
 = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES
 (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
END;
```

4. 使用如下命令测试。

   · 只读测试：

```
polar.o.x8.4xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=100000000
polar.o.x8.4xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=500000000
polar.o.x8.4xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=1000000000
polar.o.x8.2xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=100000000
polar.o.x8.2xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=500000000
polar.o.x8.2xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x8.xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 32 -j 64 -T 120 -D
scale=10000 -D range=100000000
polar.o.x8.xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 32 -j 64 -T 120 -D
scale=10000 -D range=500000000
polar.o.x8.xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 32 -j 64 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.xlarge，总数据量10亿，热数据1亿
```

```
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=100000000
polar.o.x4.xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=500000000
polar.o.x4.xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.large，总数据量5亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=100000000
polar.o.x4.large，总数据量5亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=500000000
polar.o.x4.medium，总数据量1亿，热数据5000万
pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=50000000
polar.o.x4.medium，总数据量1亿，热数据1亿
pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=100000000
```

- **读写测试：**

```
polar.o.x8.4xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=100000000
polar.o.x8.4xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=500000000
polar.o.x8.4xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D
 scale=10000 -D range=1000000000
polar.o.x8.2xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=100000000
polar.o.x8.2xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=500000000
polar.o.x8.2xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x8.xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 32 -j 64 -T 120 -D
scale=10000 -D range=100000000
polar.o.x8.xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 64 -T 120 -D
scale=10000 -D range=500000000
polar.o.x8.xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 64 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.xlarge，总数据量10亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=100000000
polar.o.x4.xlarge，总数据量10亿，热数据5亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=500000000
polar.o.x4.xlarge，总数据量10亿，热数据10亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D
scale=10000 -D range=1000000000
polar.o.x4.large，总数据量5亿，热数据1亿
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=100000000
polar.o.x4.large，总数据量5亿，热数据5亿
```

```
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 16 -j 16 -T 120 -D
scale=5000 -D range=500000000
polar.o.x4.medium，总数据量1亿，热数据5000万
pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=50000000
polar.o.x4.medium，总数据量1亿，热数据1亿
pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 8 -j 8 -T 120 -D
scale=1000 -D range=100000000
```

说明:

- scale乘以10万：表示测试数据量。
- range：表示活跃数据量。
- -c：表示测试连接数，测试连接数不代表该规格的最大连接数，最大连接数请参考#unique_5。

测试结果

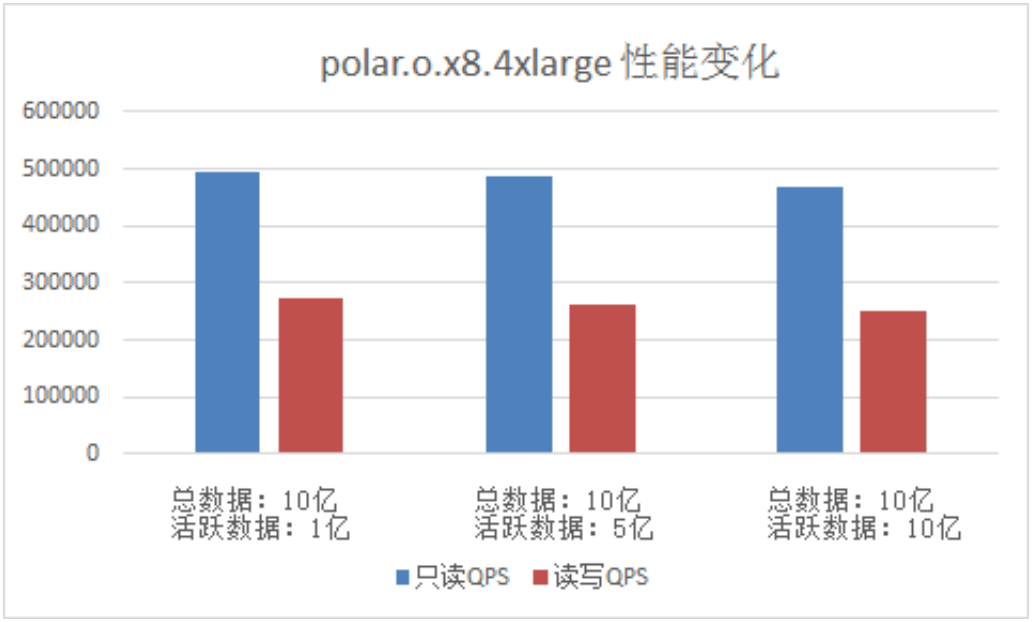
规格	测试数据量	热（活跃）数据量	只读QPS	读写QPS
polar.o.x8.4xlarge 32核 256G	10亿	1亿	522160	274270
polar.o.x8.4xlarge 32核 128G	10亿	5亿	514143	262859
polar.o.x8.4xlarge 32核 128G	10亿	10亿	493321	249679
polar.o.x8.2xlarge 16核 128G	10亿	1亿	256998	145386
polar.o.x8.2xlarge 16核 128G	10亿	5亿	253937	123806
polar.o.x8.2xlarge 16核 128G	10亿	10亿	243326	107800

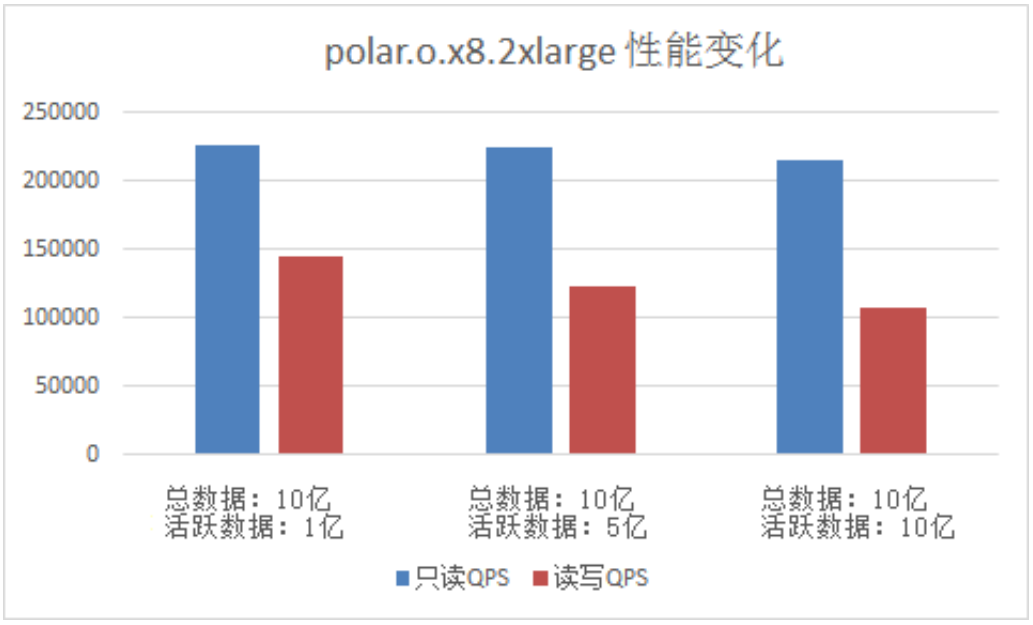
规格	测试数据量	热（活跃）数据量	只读QPS	读写QPS
polar.o.x8.xlarge 8核 64G	10亿	1亿	159323	66792
polar.o.x8.xlarge 8核 64G	10亿	5亿	155498	54070
polar.o.x8.xlarge 8核 64G	10亿	10亿	152735	54456
polar.o.x4.xlarge 8核 32G	10亿	1亿	129323	59738
polar.o.x4.xlarge 8核 32G	10亿	5亿	115498	49924
polar.o.x4.xlarge 8核 32G	10亿	10亿	102735	47946
polar.o.x4.large 4核 16G	5亿	1亿	75729	45242
polar.o.x4.large 4核 16G	5亿	5亿	63818	40308
polar.o.x4.medium 2核 8G	1亿	5000万	34386	19886
polar.o.x4.medium 2核 8G	1亿	1亿	33752	18242

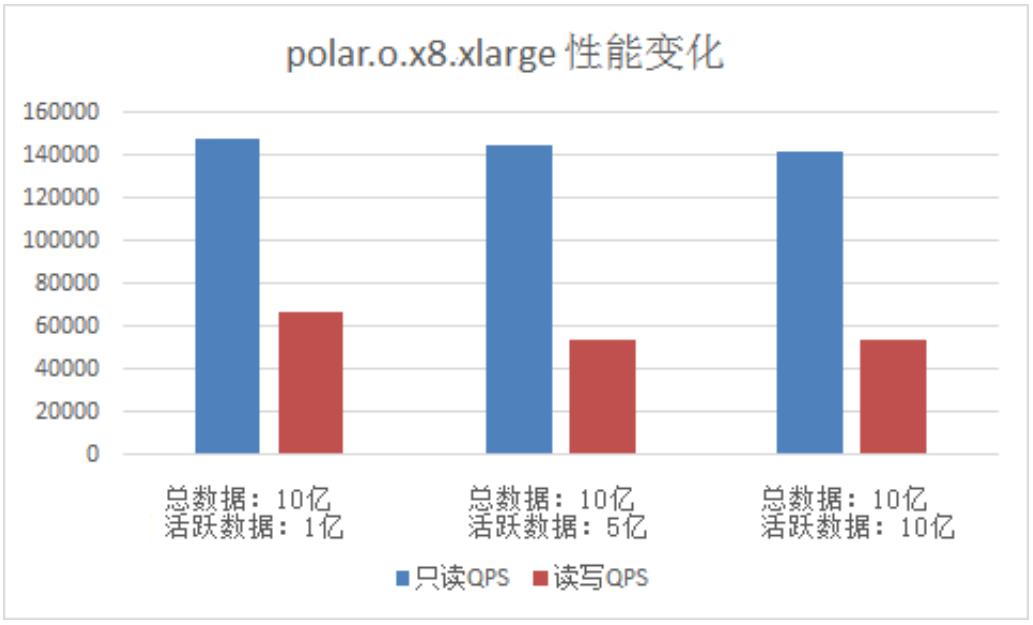
说明:

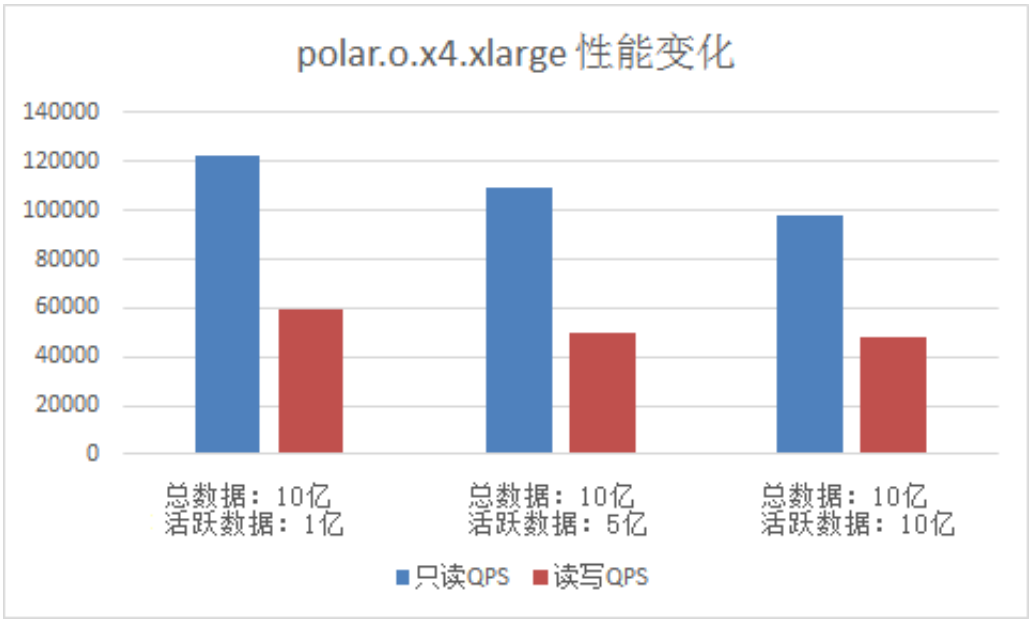
· 规格：POLARDB for Oracle的规格代码（提工单调整内存和RDS for PostgreSQL相同）。

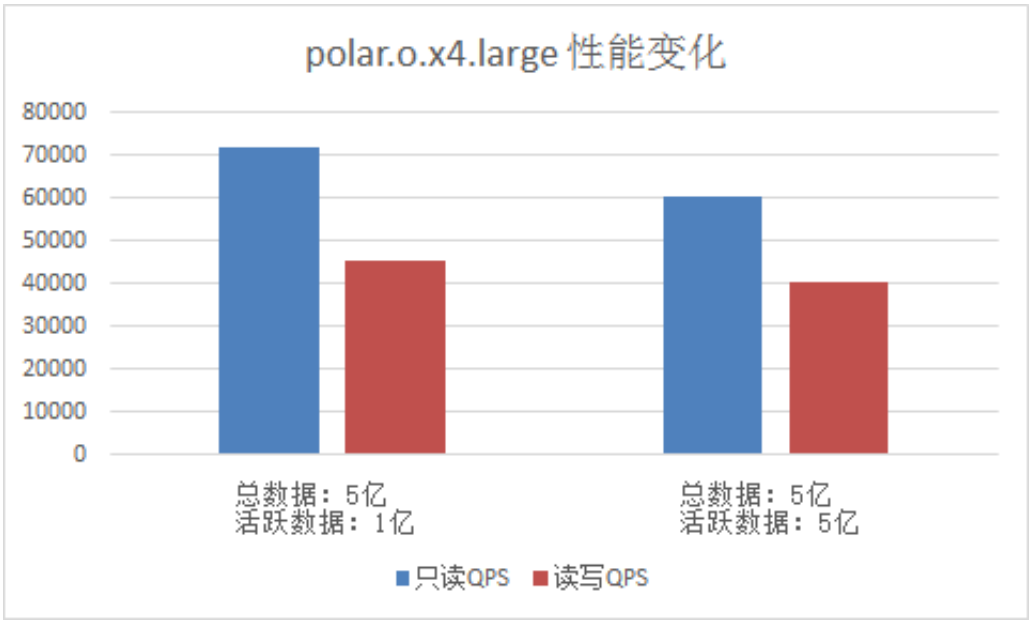
- 预计默认存储空间可存储数据量：预计该规格默认存储空间可以存储的记录条数。
- 测试数据量：本轮测试数据的记录条数。
- 热（活跃）数据量：本轮测试的查询、更新SQL的记录条数。
- 只读QPS：只读测试的结果，表示每秒请求数。
- 读写QPS：读写测试的结果，表示每秒请求数。

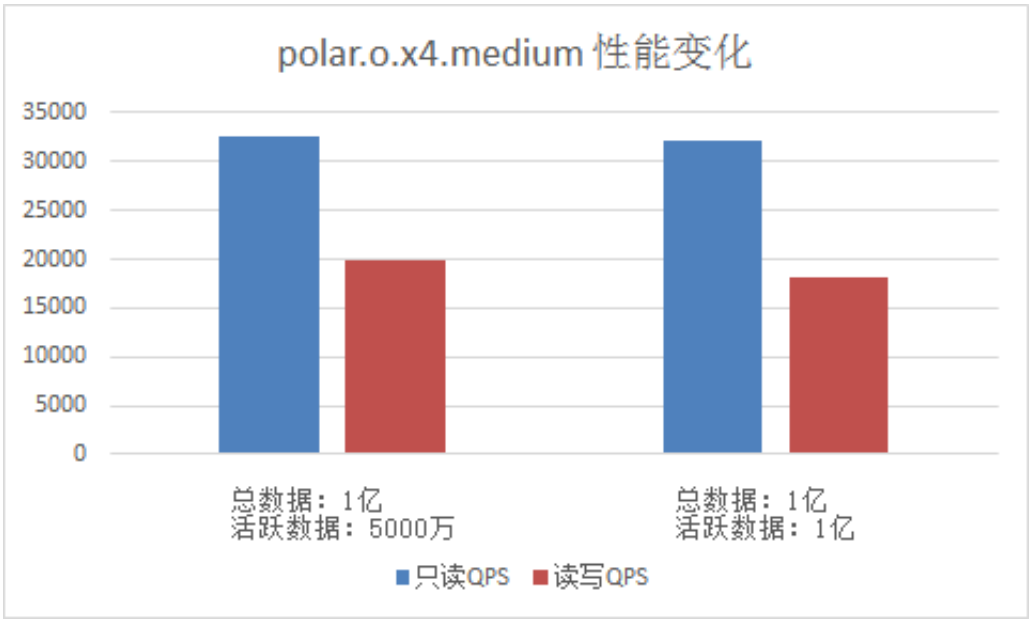





polar.o.x8.xlarge 性能变化
160000
140000
120000
100000
80000
60000
40000
20000
0
总数据：10亿
活跃数据：1亿
总数据：10亿
活跃数据：5亿
总数据：10亿
活跃数据：10亿
只读QPS
读写QPS


polar.o.x4.xlarge 性能变化
140000
120000
100000
80000
60000
40000
20000
0
总数据：10亿
活跃数据：1亿
总数据：10亿
活跃数据：5亿
总数据：10亿
活跃数据：10亿
只读QPS
读写QPS


polar.o.x4.large 性能变化
80000
70000
60000
50000
40000
30000
20000
10000
0
总数据：5亿
活跃数据：1亿
总数据：5亿
活跃数据：5亿
只读QPS
读写QPS


polar.o.x4.medium 性能变化
35000
30000
25000
20000
15000
10000
5000
0
总数据：1亿
活跃数据：5000万
总数据：1亿
活跃数据：1亿
只读QPS
读写QPS